SOFTWARE TESTING

Start

Done By:
Ammar Yasser
Ahmed ayman
Rowan mohamed mohamed
Mariam mohamed nabeh el kholy
Nadeen Elsayed

SUPERVISED BY

ENG : Abdelrahman
osama

INTRODUCTION

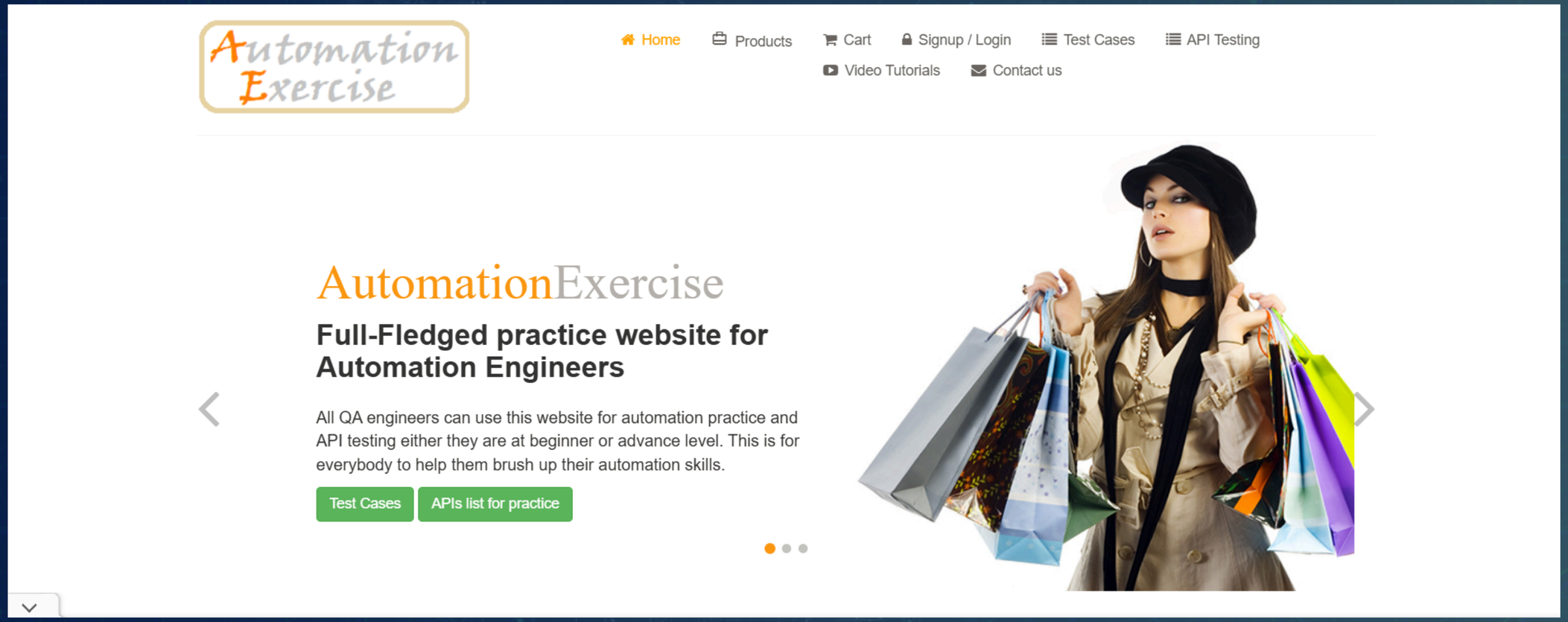
OUR WEBSITE

Automation Exercise is a practice web application designed to help learners and testers improve their automation skills. It simulates a real e-commerce site with features like user registration, login, product listings, shopping cart, and contact forms — offering a hands-on environment for practicing UI automation, API testing, and end-to-end test scenarios.

OUR WEBSITE

- Access a real-world e-commerce platform to practice UI test automation.
- Interact with features like user signup, login, product search, cart operations, and checkout flow.
- Ideal for testing frameworks like Selenium, Cypress, and TestNG.
- Enables hands-on learning of end-to-end test scenarios in a dynamic web environment.

OUR WEBSITE


Automation Exercise
Home
Products
Cart
Signup / Login
Test Cases
API Testing
Video Tutorials
Contact us
AutomationExercise
Full-Fledged practice website for Automation Engineers
All QA engineers can use this website for automation practice and API testing either they are at beginner or advance level. This is for everybody to help them brush up their automation skills.
Test Cases
APIs list for practice

WHY WE CHOOSE WEBSITE

- Clean, user-friendly interface.
- Rich feature set suitable for testing.
- Alignment with project goals (manual & automation testing)

PROJECT OBJECTIVE

- Develop an automation testing framework.
- Ensure tests are accurate, fast, and repeatable.
- Focus on key scenarios: Login and payment.

ABOUT PROJECT

ADMINISTRATOR

- manage inventory
- process orders
- handle customer support

.

ABOUT PROJECT

CUSTOMER

- browse products
- make purchases
- track orders

.

ABOUT PROJECT

SOFTWARE REQUIRMENTS SPECIFICATION

- 🔍 Identified system objectives and clearly defined the scope of the website testing project.
- 🧾 Documented functional requirements including user interactions like registration, login, and cart operations.
- 🧪 Specified non-functional requirements such as performance, usability, and browser compatibility.
- 🛠️ Outlined test cases and expected behaviors based on the system flow.
- 📄 Created detailed sections for user stories, use cases, and feature requirements.
- ✅ Ensured the SRS aligns with both manual and automated testing goals.

FUNCTIONAL REQUIRMENTS

1. user authintication: sercure login and sign up

2. product catalog: browse, search and filter products

3. cart and check out: add , remove and update items

FUNCTIONAL REQUIRMENTS

4. payment processing: secure transaction via various payment methods

5. order management: track orders and view invoices

NON-FUNCTIONAL REQUIRMENTS

1. performance
2. scalability
3. reliability

NON-FUNCTIONAL REQUIRMENTS

4. system availability

5. useability

6. security

ABOUT PROJECT

MANUAL TESTING

- Performed end-to-end manual testing of core features including user registration, login, product search, and cart flow.
- Identified UI/UX issues and validated form inputs under different scenarios.
- Logged bugs and edge cases using structured test cases and real-time observations.
- Helped ensure functionality before automation implementation.

TEST CASES

SC02		Add multiple items to the cart	user must register as a customer	1. Open the app 2. Select multiple items 3. Click "Add to Cart" for each	many items from products	All selected items appear in the cart	item is added successfully
SC03		Add the same item multiple times	user must register as a customer	1. Open the app 2. Select an item 3. Click "Add to Cart" multiple times	add one product from the products added before	The item's quantity increases correctly	item is added successfully
SC04		Verify cart updates after adding	user must register as a customer	1. Add an item to the cart 2. Open the cart		The correct item & quantity appear in the cart	verfied correctly
SC05		Add an item when the cart is empty	user must register as a customer	1. Ensure the cart is empty 2. Add an item	item from products	The cart updates with the selected item	item is added successfully
SC06	removing item from cart	Remove a single item from the cart	user must register as a customer	1. Add an item to the cart 2. Click "Remove"		The item is removed from the cart	removed correctly
SC07		Remove multiple items from the cart	user must register as a customer	1. Add multiple items to the cart 2. Remove one item at a time		The selected items are removed, others remain	removed correctly
SC08		Remove all items (empty the cart)	user must register as a customer	1. Add multiple items 2. Click "Remove" on each item		The cart becomes empty	removed correctly
SC09		Verify cart updates after removal	user must register as a customer	1. Add an item 2. Remove it 3. Check the cart		The cart no longer contains the removed item	verfied correctly
SC10		Remove an item that was added multiple times	user must register as a customer	1. Add an item multiple times 2. Remove it once		The quantity decreases instead of complete removal	removed correctly
SC11	refresh the page	Add an item and refresh the page	user must register as a customer	1. Add an item 2. Refresh the page 3. Check the cart	add one product to the cart	The item should still be in the cart	item is still in the cart
			user must register as a	1. Remove an item 2. Refresh the page			

ABOUT PROJECT

AUTOMATION TESTING

- Automated critical user flows using Selenium WebDriver and managed test suites with TestNG.
- Focused on verifying functionality, input validation, and flow consistency.
- Simulated real-user interactions like adding to cart, form submissions, and navigation.
- Enabled faster regression testing and enhanced test coverage.

PAGE OBJECT MODEL(POM)

POM is a design pattern used in test automation frameworks (commonly with tools like Selenium) to enhance code reusability, readability, and maintenance.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    <groupId>org.example</groupId>
    <artifactId>Phptravels</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>23</maven.compiler.source>
        <maven.compiler.target>23</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <!-- Selenium Java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
        </dependency>
        <!-- WebDriverManager (to manage ChromeDriver automatically) -->
        <dependency>
            <groupId>io.github.bonigarcia</groupId>
            <artifactId>webdrivermanager</artifactId>
            <version>5.6.2</version>
        </dependency>
        <!-- TestNG -->
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>6.14.3</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

studio shodwe

TEST SCRIPT

```java
public class TestCases_PositiveSenario extends TestBase {
    HomePage homeObject = new HomePage(driver);
    TestCasesPage TestCasesObject= new TestCasesPage(driver);

    @Test (priority = 1)
    public void testCases_VerifyPage() throws InterruptedException {
        Assert.assertEquals("rgba(255, 165, 0, 1)",homeObject.homeLink.getCssValue(propertyName:"color"));
        homeObject.openTestCasespPage();
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(seconds:10));
        wait.until(ExpectedConditions.visibilityOf(TestCasesObject.TestCasesMessage));
        Assert.assertEquals("TEST CASES", TestCasesObject.TestCasesMessage.getText());
    }
}
```

TEST SCRIPT

```java
public class ContactUs_happyScenario extends TestBase {
    HomePage homeObject = new HomePage(driver);
    ContactUsPage contactUsObject = new ContactUsPage(driver);
    String name = LoadContactUsDate.userData.getProperty(key:"name");
    String email = LoadContactUsDate.userData.getProperty(key:"email");
    String subject = LoadContactUsDate.userData.getProperty(key:"subject");
    String message = LoadContactUsDate.userData.getProperty(key:"message");
    String path = LoadContactUsDate.userData.getProperty(key:"path");
  @Test
  public void testContactUs_validData() throws InterruptedException, AWTException {
      homeObject.openContactUsPage();
      WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(seconds:10));
      wait.until(ExpectedConditions.visibilityOf(contactUsObject.getInMessage));
      Assert.assertEquals("GET IN TOUCH", contactUsObject.getInMessage.getText());
      contactUsObject.userCanContactUs(name,email,subject,message,path);
      wait.until(ExpectedConditions.visibilityOf(contactUsObject.successMessage));
      Assert.assertEquals("Success! Your details have been submitted successfully.", contactUsObject.successMessage.getText());
  }
}
```

Loading...

TOOLS AND ENVIRONMENT

TOOLS USED IN PROJECT

Selenium Webdriver

- Open-source, widely used
- Supports multiple browsers (Chrome, Firefox, etc.)
- Integrates with other tools (like TestNG, Maven, CI tools)

TOOLS USED IN PROJECT

TestNG

- Organizes test flow using annotations (@Test, @BeforeClass, etc.)
- Allows grouping and prioritizing test cases
- Supports parallel test execution for faster results
- Generates HTML reports automatically
- Enables data-driven testing using
- Easily integrates with Selenium, Maven

TOOLS USED IN PROJECT

eclipse

- ✔ Strong Java support - Ideal for Selenium and TestNG, which are Java-based.
- 🛠 Built-in project structure - Makes organizing test files easy and efficient.
- 🧠 Code assistance - Features like auto-complete and quick fixes speed up development.
- 🐞 Powerful debugger - Helps in tracking and fixing automation bugs easily.
- 🔌 TestNG integration - Smooth setup and execution of automated test suites.
- 🌐 Widely used - Well-documented and trusted in the testing community.

TOOLS USED IN PROJECT

Maven

- Manages project dependencies automatically (like Selenium and TestNG jars)
- Defines project structure and build process
- Makes it easy to run tests and build reports using plugins
- Ensures consistent setup across different machines
-

PROJECT OVERVIEW

1. ✅ Performed manual testing on key features like registration, login, product browsing, and cart workflows.

2. 🐛 Documented bugs using structured test cases and reported UI issues, form errors, and flow inconsistencies.

3. Used Selenium WebDriver for browser automation

PROJECT OVERVIEW

4. Managed tests with TestNG and use MAVEN for dependencies

5. Developed and executed tests in eclipse IDEA

6. Validated both valid and invalid user scenarios to ensure system reliability

PROJECT OVERVIEW

7.

- 🧪 Built reusable test scripts for critical flows like user login, product filtering, and checkout.
- r.

8.

⏱️ Improved testing speed, coverage, and accuracy through automation.

ANY QUESTION ?

THANK YOU!

we hope you enjoyed the presentation and find it helpful.